

Table 1 (cont')

```

for (nn = mn = 0, more = 1; more;) {
    for (i = more = 0; i < 2; i++) {
        /*
         * do we have more of this sequence?
         */
        if (!*ps[i])
            continue;
        more++;
        if (pp[i].spc) { /* leading space */
            *po[i]++ = ' ';
            pp[i].spc--;
        }
        else if (siz[i]) { /* in a gap */
            *po[i]++ = '-';
            siz[i]--;
        }
        else { /* we're putting a seq element
            */
            *po[i] = *ps[i];
            if (islower(*ps[i]))
                *ps[i] = toupper(*ps[i]);
            po[i]++;
            ps[i]++;
            /*
             * are we at next gap for this seq?
             */
            if (ni[i] == pp[i].x[ij[i]]) {
                /*
                 * we need to merge all gaps
                 * at this location
                 */
                siz[i] = pp[i].n[ij[i] + +];
                while (ni[i] == pp[i].x[ij[i]])
                    siz[i] += pp[i].n[ij[i] + +];
            }
            ni[i]++;
        }
    }
    if (++nn == olen || !more && nn) {
        dumpblock();
        for (i = 0; i < 2; i++)
            po[i] = out[i];
        nn = 0;
    }
}

/*
 * dump a block of lines, including numbers, stars: pr_align()
 */
static
dumpblock()
{
    register i;

    for (i = 0; i < 2; i++)
        *po[i]-- = '\0';
}

```

...pr_align

dumpblock

Table 1 (cont')**...dumpblock**

```

5      (void) putc('\n', fx);
      for (i = 0; i < 2; i++) {
          if (*out[i] && (*out[i] != ' ' || *po[i] != ' ')) {
              if (i == 0)
                  nums(i);
              if (i == 0 && *out[1])
                  stars();
10         putline(i);
              if (i == 0 && *out[1])
                  fprintf(fx, star);
              if (i == 1)
                  nums(i);
15     }
    }
}

/*
20  * put out a number line: dumpblock()
  */
static
nums(ix)                                nums
25  {
    int      ix;      /* index in out[] holding seq line */
    char      nline[P_LINE];
    register  i, j;
    register  char *pn, *px, *py;

30     for (pn = nline, i = 0; i < lmax+P_SPC; i++, pn++)
        *pn = ' ';
    for (i = nc[ix], py = out[ix]; *py; py++, pn++) {
        if (*py == ' ' || *py == '-')
            *pn = ' ';
35         else {
            if (i%10 == 0 || (i == 1 && nc[ix] != 1)) {
                j = (i < 0)? -i : i;
                for (px = pn; j /= 10, px--)
                    *px = j%10 + '0';
40                 if (i < 0)
                    *px = '-';
            }
            else
                *pn = ' ';
45             i++;
        }
    }
    *pn = '0';
    nc[ix] = i;
50     for (pn = nline; *pn; pn++)
        (void) putc(*pn, fx);
    (void) putc('\n', fx);
}

55  /*
  * put out a line (name, [num], seq, [num]): dumpblock()
  */
static
putline(ix)                                putline
60  {
    int      ix;
    {

```

Table 1 (cont')

...putline

```
5      int          i;
      register char *px;

      for (px = namex[ix], i = 0; *px && *px != ':'; px++, i++)
          (void) putc(*px, fx);
      for (; i < lmax+P_SPC; i++)
          (void) putc(' ', fx);

10     /* these count from 1:
       * ni[] is current element (from 1)
       * nc[] is number at start of current line
       */
15     for (px = out[ix]; *px; px++)
          (void) putc(*px&0x7F, fx);
          (void) putc('\n', fx);
    }

20     /*
       * put a line of stars (seqs always in out[0], out[1]): dumpblock()
       */
      static
25     stars()
    {
        int          i;
        register char *p0, *p1, cx, *px;

30         if (!*out[0] || (*out[0] == ' ' && *(p0[0]) == ' ') ||
            !*out[1] || (*out[1] == ' ' && *(p0[1]) == ' '))
            return;
        px = star;
        for (i = lmax+P_SPC; i; i--)
35             *px++ = ' ';

        for (p0 = out[0], p1 = out[1]; *p0 && *p1; p0++, p1++) {
            if (isalpha(*p0) && isalpha(*p1)) {
40                 if (xbm[*p0-'A']&xbm[*p1-'A']) {
                     cx = '*';
                     nm++;
                 }
45                 else if (!dna && _day[*p0-'A'][*p1-'A'] > 0)
                     cx = '.';
                 else
                     cx = ' ';
            }
            else
50                 cx = ' ';
            *px++ = cx;
        }
        *px++ = '\n';
        *px = '\0';
55     }
```

stars